LPR_2.0: Mejoras a un ambiente de aprendizaje de servicios de red.

Yrigaray Elian[1], Barrionuevo Mercedes[1], Taffermaberry Carlos[12]
1 Universidad Nacional de San Luis,
Ejército de los Andes 950 - 5700 - San Luis - República Argentina
2 Universidad Tecnológica Nacional, Facultad Regional Mendoza
Rodriguez 273, Ciudad Mendoza -M5502AJE - República Argentina
{ mdbarrio, ctaffer}@unsl.edu.ar
{elian10292}@gmail.com

Resumen

Uno de los objetivos de la virtualización es la reproducción completa de una red de computadoras físicas en software, pudiendo trabajar con dispositivos y servicios de red lógicos. Ayudar a la metodología de enseñanza tradicional mediante la experimentación, usando un laboratorio virtual y portátil, permite realizar pruebas en cualquier momento y lugar. LPR (Laboratorio Portátil de Redes) es una herramienta que se creó con el fin de construir un ambiente virtualizado de redes de computadoras permitiendo conectar, configurar y comunicar máquinas clientes y servidores. En este trabajo se presentan mejoras que permiten flexibilizar y ampliar su alcance mediante la incorporación de un nuevo módulo gráfico para el diseño de diversas topologías de red.

Palabras clave: Virtualización. Servicios de Red. Software Libre. Educación. Interfaces de usuario.

1 Introducción

En la actualidad, la sociedad global interactúa con Internet a través del uso de distintas aplicaciones como World Wide Web, redes sociales, programas que permiten compartir información, etc., sin la necesidad de conocer realmente las tecnologías y protocolos subyacentes. Sin embargo, para aquellos interesados en el estudio del comportamiento de las redes de datos, se vuelve cada vez más importante conocer cómo un servicio de red es capaz de transmitir e interpretar los mensajes enviados y recibidos a través de la red.

Es fundamental para todo estudiante de Tecnologías de la Información y la Comunicación (TIC) poder configurar y probar el correcto funcionamiento de los servicios más comunes utilizados en Internet. Realizar diversas pruebas en un ambiente seguro, donde se pueda volver a una configuración inicial y estable, es un objetivo de toda cátedra dedicada a la puesta a punto de los servicios de red.

Permitir a diversas aplicaciones ejecutarse en una red virtual exactamente igual que en una red física es un logro de la virtualización de redes. Algunos entornos de virtualización permiten realizar diversas pruebas de configuración y ejecución de comandos o de aplicaciones pudiendo guardar el estado actual y recuperar el trabajo realizado luego de un tiempo.

En base a lo mencionado anteriormente, fue creado el Laboratorio Portátil de Redes (LPR) [1], una herramienta fundamental para el desarrollo de las prácticas en las materias Servicios en SO de Redes de la Tecnicatura Universitaria en Redes de Computadoras (TUR) de la Universidad Nacional de San Luis (UNSL).

El presente trabajo describe una mejora realizada a LPR, la cual anexa un módulo gráfico que le proporciona al usuario la posibilidad de diseñar distintas topologías de red con el modo WYSIWYG (acrónimo de

What You See Is What You Get). A partir de estas topologías es posible desplegar el escenario a simular utilizando virtualización, funcionalidad que ya estaba incluida en LPR.
El objetivo perseguido fue lograr más flexibilidad a la hora de establecer configuraciones en los distintos dispositivos y diseños de redes mediante un entorno visual fácil de usar, de modo que fluya la comunicación entre el usuario y la herramienta, haciendo uso de las gramáticas visuales y verbales como: iconos, botones, menús, tipografía, entre otros.
Durante el desarrollo del presente trabajo se evaluaron distintos tipos de herramientas de diseño de interfaces gráficas de usuario, analizando cuál de ellas era la más eficiente y de menor consumo de hardware. Esto último es con el afán de complementar a LPR para el desarrollo no sólo de las prácticas correspondientes a la asignatura Servicios en Sistemas Operativos de Redes sino también, de todas aquellas materias que lo requieran, colaborando así con docentes y estudiantes en el proceso de enseñanza-aprendizaje.
En las siguientes secciones se detallan los conceptos teóricos involucrados en este trabajo, las mejoras propuestas al Laboratorio Portátil de Redes, diversas pruebas y ensayos realizados y, finalmente, conclusiones y trabajos futuros.

2 Marco Teórico

Esta sección introduce diferentes conceptos utilizados en el desarrollo de nuestro trabajo, destacándose: la virtualización y las interfaces gráficas de usuarios. A continuación, se enumeran las características principales de cada uno de ellos.

2.1 Virtualización

La virtualización es la abstracción de los recursos de una computadora y su puesta en funcionamiento como máquina virtual en otra máquina física [2].

Normalmente, cuando se utiliza una computadora, el Sistema Operativo (SO) es instalado y ejecutado directamente sobre el hardware y de esta forma se aprovecha de forma completa su potencial. Sin embargo, existe un tipo de programa llamado software de virtualización o hipervisor, encargado sólo de multiplexar el hardware, aprovechando los recursos reales de la computadora y, sobre este software, es posible instalar distintos SO como si se tratase de una computadora real. El SO huésped funciona, a grandes rasgos, como si fuera un SO principal, por lo que todas sus funciones y características están disponibles, convirtiéndolo en una herramienta ideal para realizar pruebas [3].
Cuando hablamos de virtualización hacemos referencia al proceso de reemplazar dispositivos físicos por dispositivos virtuales, disponibles mediante el uso de un software. Se pueden virtualizar servidores, estaciones de trabajo, redes y aplicaciones. Para ello, el software de virtualización administra los recursos físicos de esa máquina: memoria, CPU, almacenamiento y ancho de banda de la red, entre los aspectos más relevantes.
Junto con la disminución del uso de equipos físicos, la virtualización trae como beneficios la reducción de costos de mantenimiento y consumo energético. Esto deriva una consolidación de servidores, optimizando el uso del espacio físico.
Cuando se trata de la virtualización, existen varias maneras de lograr el mismo resultado a través de diferentes técnicas, siendo las más utilizadas: *Paravirtualización, Nivel de SO, Virtualización completa, Nivel de Kernel* [2, 4, 5], entre otros.
En la figura 1 se pueden observar los tipos y tecnologías para realizar virtualización y el lugar que ocupa LPR dentro de esta clasificación.

Figura 1: Tipos de Virtualización y contexto de LPR.

La tecnología de principal interés para la creación de LPR fue *Docker* [6], esto es como consecuencia de ser una herramienta de software libre que virtualiza en Nivel SO, además de permitir empaquetar entornos y aplicaciones que posteriormente se pueden desplegar en otro SO con esta tecnología y permitir el acceso a la virtualización del kernel de Linux a través de la biblioteca *libcontainer* [7], o indirectamente a través de *LXC* [8].

La fundamentación de elección de esta herramienta se desarrolló en base a los estudios y pruebas realizadas en [1] dando origen, hace un año atrás a LPR, un ambiente de trabajo donde se logró interconectar máquinas clientes y servidores de diversos servicios de red a nivel de capa de aplicación en una misma estación de trabajo de forma virtualizada.

No solamente es necesario contar con un software de virtualización para administrar servicios de red sino también se necesita trabajar con herramientas de diseño de topologías de red que permitan crear esquemas de redes en base a las necesidades del usuario. Por lo tanto surgió la necesidad de adicionar a LPR una herramienta de código libre para diseñar de forma gráfica y sencilla diferentes topologías de red a simular.

A continuación se detallan diversos aspectos a considerar a la hora de desarrollar una herramienta gráfica tales como: interfaces gráficas de usuarios, tipos, herramientas de código libre existentes, entre otras.

2.2 Interfaces gráficas de usuarios.

El rol principal de la interfaz gráfica de usuario, también denominada GUI (Graphical User Interface), es hacer de intermediario entre el usuario y la computadora. Es un software que muestra de forma visual todas las acciones posibles en una plataforma, así como la información disponible, de modo que los usuarios puedan interactuar con mayor facilidad, sin necesidad de disponer de profundos conocimientos de informática.

Las interfaces gráficas son un tipo de interfaz de usuario que utiliza imágenes, iconos y menús para mostrar las acciones disponibles que el usuario puede seleccionar en un dispositivo. Estas aparecen con mayor o menor frecuencia en diferentes tipos de dispositivos, contando con ciertas peculiaridades que las diferencian entre ellas. Algunos tipos de interfaces gráficas de usuario existentes en la actualidad son:

- *GUI (Grafical User Interface)* [9]*:* La principal finalidad de este programa es simplificar y garantizar una cómoda interacción entre una persona y un dispositivo, es decir, permitir una buena accesibilidad, posibilitando un mayor aprovechamiento de la tecnología.
- *PUI (Perceptual User Interface)* [10]*:* Las interfaces perceptuales de usuario utilizan las capacidades humanas con el fin de poder crear interfaces más naturales e intuitivas. Habilitan múltiples estilos de interacción, como por ejemplo: únicamente voz, voz y texto, texto y tacto, visión y sonido. Estas vías de interacción son usadas apropiadamente en diferentes circunstancias, con el objetivo de hacer la interacción más sencilla. Las PUI, utilizan para su desarrollo las capacidades perceptivas humanas, de modo que pueden presentar la información y el contexto en forma natural y significativa.

- *Touchscreen* [9]*:* Algunas GUI son diseñadas para cumplir con los rigurosos requisitos de los mercados verticales, entre las cuales se encuentra la Touchscreen o GUI de uso específico, una pantalla que al ser tocada efectúa los comandos del ratón en el software. Esta interfaz fue diseñada por Gene Mosher en la computadora del ST de Atari en 1986. El uso que especificó en las GUI de pantalla táctil ha encabezado una revolución mundial en el uso de las computadoras a través de las industrias alimenticias y de bebidas, y en ventas al por menor.

De estos tipos detallados anteriormente, la mejor opción para el desarrollo de nuestro trabajo son las interfaces GUI. En base a esto, se evaluaron algunas herramientas existentes de código libre, las cuales se detallan en la próxima sección.

2.3 Herramientas para la creación de interfaces GUI.

Actualmente existen varias herramientas y bibliotecas para diseñar diversas redes de computadoras. Algunas de ellas son:

- *PyQt [11]:* implementa la popular biblioteca Qt, lo cual abre la posibilidad de desarrollar aplicaciones en Python que tengan un aspecto familiar en muchas plataformas, al mismo tiempo que aprovechan las herramientas y el conocimiento de la gran comunidad de Qt. El instalador de PyQt viene con una herramienta de creación de GUI llamada Qt Designer, la cual consta de una sencilla interfaz de arrastrar y soltar, que nos permite construir rápidamente una interfaz GUI sin tener que escribir el código.
- *Tkinter [12]:* Viene incluido con Python en su versión para Windows, Mac y la mayoría de las distribuciones GNU/Linux. Se le considera el estándar de facto en la programación GUI con Python. Es fácil de usar, multiplataforma, es un binding de la biblioteca Tcl/Tk que está también disponible para otros lenguajes como Perl y Ruby.
- *GNS3 (Graphic Network Simulation) [13]:* es un simulador gráfico de red que permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellos. Con GNS3 los usuarios tienen la posibilidad de poder escoger cada uno de los elementos que llegarán a formar parte de una red informática. GNS3 está estrechamente vinculada con: Dynamips, un emulador de IOS que permite a los usuarios ejecutar binarios imágenes IOS de Cisco Systems; Dynagen, un front-end basado en texto para Dynamips y Qemu, un emulador de PIX.
- *DIA [14]*: es una aplicación de código libre usada para diseño y edición gráfica de diagramas. Esta herramienta se puede usar para dibujar diferentes tipos de diagramas, como: diagramas UML, de relaciones de entidades, de flujo, de red, entre otros. También es posible agregar soporte para nuevas formas escribiendo archivos XML simples, utilizando un subconjunto de SVG para dibujar la forma.
- *KivaNS (Kiva Network Simulator) [15]:* es una aplicación gratuita y de código abierto basada en Java para especificar esquemas de redes de datos y simular el enrutamiento de paquetes a través de tales redes. Tiene como objetivo principal ayudar a diseñar y comprender el funcionamiento de redes de datos, y en especial el encaminamiento de paquetes en la arquitectura TCP/IP, sin necesidad de una infraestructura real y de herramientas de análisis de tráfico. Esta herramienta fue desarrollada por la Universidad de Alicante (España) como un trabajo de tesis orientada a la formación académica. KivaNS también es capaz de simular distintos tipos de

errores en el funcionamiento de las redes, como la pérdida de paquetes o fallos en tablas de enrutamiento.

En el apartado 3.3 se analizarán ventajas y desventajas de las distintas herramientas, seleccionando la más adecuada para el diseño de la interfaz GUI para el módulo de creación de topologías de red.

3 Mejoras al Laboratorio Portátil de Redes

LPR_2.0 es una herramienta que combina servicios de SO basados en GNU/Ubuntu y virtualización con Docker sumado al diseño de topologías de red de forma flexible y amigable al usuario.
La arquitectura de LPR_2.0 se muestra en la figura 2. En ella se puede observar su arquitectura formada por dos módulos independientes y bien diferenciados: *LPR y EliaNS.*

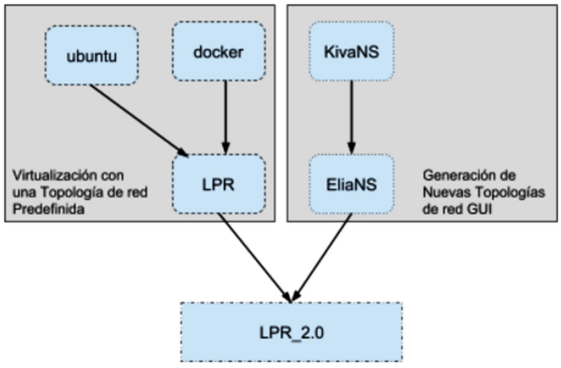

Figura 2: Arquitectura de LPR_2.0

3.1 LPR

La herramienta LPR está siendo utilizada como soporte para las prácticas en la materia "Servicios en Sistemas Operativos de Redes" de la Tecnicatura Universitaria en Redes de Computadoras de la UNSL. En dicho ámbito, los estudiantes llevan a cabo la configuración, puesta en marcha y verificación del correcto funcionamiento de distintos servidores, tales como: DNS, DHCP, HTTP, SMTP, entre otros.
LPR es una herramienta de software libre y disponible para su descarga desde el sitio web del Departamento de Informática de la UNSL [16], permite desplegar virtualmente la topología mostrada en la Figura 3 y a partir de ella configurar los distintos tipos de servicios de red, iniciando contenedores de Docker.
Para un manejo más sencillo de la orquestación de la topología, en cada uno de los prácticos fueron utilizados distintos scripts de inicio, pausa, reanudación y finalización de Docker para activar, detener, reanudar y terminar respectivamente las máquinas virtuales necesarias para cada servicio.

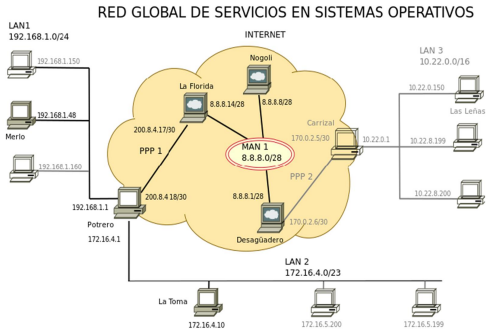

Figura 3: Topología de red existente.

En su última versión la herramienta posee un tamaño de 1,23 Gigabytes y para su correcto funcionamiento, los requisitos de hardware son: procesador Intel o AMD x64 y al menos 4 Gigabytes de memoria RAM. Esto permite que los alumnos puedan llevar a cabo favorablemente las prácticas en el Laboratorio de Redes del Departamento de Informática de la UNSL, como así también en su hogar si disponen de una computadora que cumpla con los requisitos mencionados anteriormente.
Si bien la herramienta cumple con las necesidades y requerimientos de la materia, en su versión inicial presenta la limitación de admitir una única topología de red a virtualizar, situación que se desea modificar para tener un mayor alcance y poder servir a otros ámbitos que lo requieran.

Este aspecto abordado da origen a LPR_2.0, motivo de esta publicación.

3.2 EliaNS

EliaNS es un proyecto de software libre basado en KivaNS, se encuentra disponible en una plataforma de desarrollo colaborativo de software para alojar proyectos de forma pública denominada *GitHub* [17].
EliaNS cumple con las funciones de diseño de topologías de red de forma gráfica, establecimiento de configuraciones de dispositivos de red y simulación de redes de computadoras.
Los objetivos perseguidos y logrados para la creación de este nuevo módulo fueron:

- Realizar un diseño flexible y variado de topologías de redes de computadoras.
- Establecer de manera visual:
  - ✓ La configuración de interfaces.
  - ✓ La configuración de direccionamiento IP.
  - ✓ La asignación de nombres a dispositivos e interfaces.
  - ✓ La inicialización, pausado, reanudación y finalización de contenedores Docker desde la aplicación.
- Permitir el almacenamiento persistente de la configuración de los proyectos.
- Modificar y/o agregar configuraciones a proyectos almacenados previamente.
- Optimizar el uso de recursos con el fin de que la aplicación pueda ser utilizada en computadoras con pocos requerimientos de hardware como: tamaño de memoria, de disco, etc.
- Mantener la licencia de software libre de LPR.

Usando como criterio de selección los anteriores objetivos, se analizaron ventajas y desventajas de las herramientas mencionadas en el ítem 2.3 para establecer cuál de ellas era la que mejor se adaptaría a los objetivos planteados.

3.3 Comparativa de Tecnologías para el desarrollo de EliaNS

Para poder realizar una comparación entre las diversas herramientas que permitan la creación de una interfaz simple y amigable al usuario, se instalaron varias y distintas herramientas, se desarrollaron interfaces prototipo sencillas para evaluar: complejidad de instalación y de programación, flexibilidad para realizar cambios, consumo de hardware y de software, espacio en disco, entre otros.

De la experimentación con ellas, se construyó la Tabla 1, que expresa las fortalezas y debilidades de cada una de las tecnologías consideradas.

Herramienta	Características	Ventajas	Desventajas
PyQt	-Licencia comercial y GLP. -Posibilita el desarrollo de Aplicaciones en Python.	-Consta de una herramienta de creación de GUI, llamada QtDesigner.	-Es necesario contar con conocimiento profundo sobre la gran cantidad bibliotecas y clases que posee. - No es una IDE, por lo tanto presenta dificultades para depurar y construir aplicaciones
Tkinter	-Licencia GLP. -Posibilita el desarrollo de aplicaciones en Python.	-Gran cantidad de recursos (Libros, códigos, comunidad de usuarios), que pueden ayudar. - Herramienta muy flexible a la	-Está más enfocada a la creación de interfaces sencillas como formularios,

		hora del diseño de una interfaz gráfica	calendarios, etc.
GNS3	-Licencia GNU. -Simulador gráfico de red. -Desarrollada con PyQt.	-Permite diseñar topologías de red complejas y poner en marcha simulaciones sobre ellas.	-Complejidad a la hora de su configuración y ejecución. - Requiere muchos recursos de hardware.
DIA	-Licencia GNU. -Posibilita la creación de muchos tipos de diagramas. - Desarrollado en lenguaje C.	-Brinda muchas posibilidades a la hora del diseño de topologías de redes.	- Posee muchos tipos de diagramas y funcionalidades difíciles de eliminar.
KivaNS	-Licencia GNU. -Simulador gráfico de redes de computadoras. -Orientado a simular el comportamiento del protocolo IP. -Basado en Java.	-Permite especificar esquemas de redes de datos. -Permite simular el enrutamiento de paquetes a través de código fuente simple y acotado. -Permite controlar errores de tipeo de direcciones antes de ejecutar la simulación. -Configuraciones sencillas.	-Diseño gráfico limitado debido a su falta de actualización.

Tabla 1: Comparativa de Herramientas

De acuerdo a las características detalladas, la herramienta que mejor cubrió las expectativas para los objetivos propuestos fue el entorno de desarrollo KivaNS, considerándolo como punto de partida.
Los pasos que dieron origen a EliaNS fueron:

- Identificación de clases esenciales como así también aquellas innecesarias. Las clases reutilizadas fueron: Simulador, frameInternoDibujo, Visual y Dispositivos.
- Eliminación de iconos de dispositivos innecesarios, tales como puentes, módems, topologías token ring, etc.
- Eliminación de botones correspondientes a la simulación de envíos de paquetes. Sin embargo, no fue posible eliminar por completo las clases pertenecientes a simulación, debido a que algunas de estas realizaban ciertas tareas necesarias para los requerimientos de nuestro proyecto, como por ejemplo: asignación de nombres y direcciones IP a los dispositivos e interfaces de red, creación de nuevas interfaces en los dispositivos, asignación de tablas de ruteo, etc.
- Creación en la barra de tareas de los botones de inicio, pausa, reanudación y finalización de la ejecución del proyecto en Docker, invocando a sus respectivos shell scripts.
- Validación en las ventanas de configuración sobre topologías no creadas, configuraciones en los dispositivos no cargadas o incorrectas, etc.

4 Ensayos y pruebas

La versión definitiva de la aplicación EliaNS, ocupa 1.59MB de espacio en disco, consume poca memoria RAM y bajo porcentaje de uso de CPU, como se puede observar en la figura 4. Para poder ejecutar dicha aplicación sólo es necesario tener instalada la máquina virtual de Java.

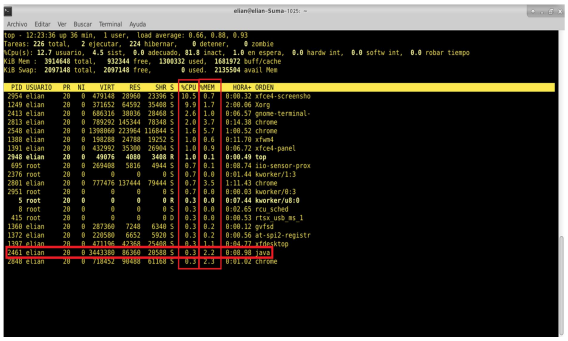

Figura 4: consumo CPU y memoria RAM

En esta sección se detallan algunos ejemplos de diseños de topologías realizadas con EliaNS.

La construcción de manera gráfica de una red local, como se observa en la Figura 5, su posterior almacenamiento, y el despliegue usando máquinas virtuales, permite verificar la flexibilidad de la herramienta EliaNS en cuanto al diseño de redes y la integración con LPR.

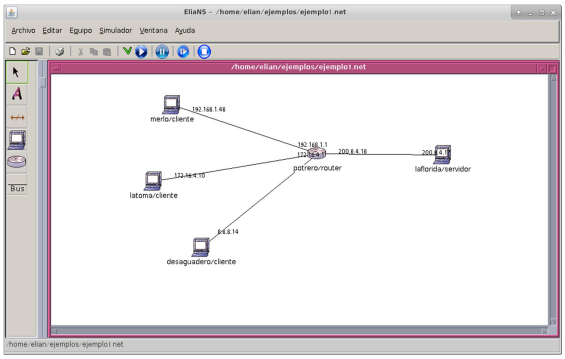

Figura 5: Topología LAN

EliaNS también permite recrear redes WAN como lo muestra la Figura 6.

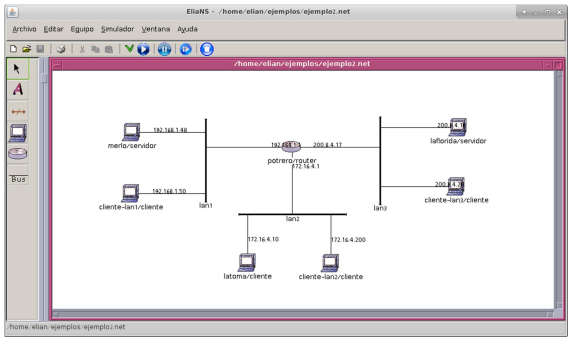

Figura 6: Topología WAN

Para poder lograr esta red WAN se seleccionaron los dispositivos que formaran parte de la red a simular, como así también, los parámetros de configuración de las distintas interfaces de red. Las Figuras 7 y 8 muestran la forma de modificar la información de una interfaz, en este caso la interfaz denominada lan1.

Figura 7: Modificación de datos de la Interfaz lan1

Figura 8: Modificación de ruta de la interfaz lan1

Finalmente, antes de generar los contenedores para comenzar a trabajar, se recomienda realizar una validación de datos, lo cual permite asegurarnos que los parámetros de la red han sido correctamente configurados permitiendo una interconexión satisfactoria.

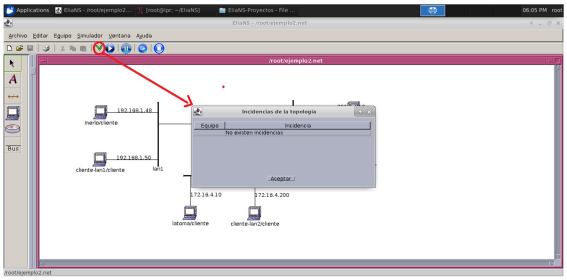

Figura 9: Validación de datos

Luego de diseñar estas topologías, oprimiendo el botón iniciar genera los contenedores docker correspondientes a cada uno de los nodos representados por EliaNS. La Figura 10 muestra la generación y ejecución de contenedores Docker, correspondientes a la red de la Figura 6.

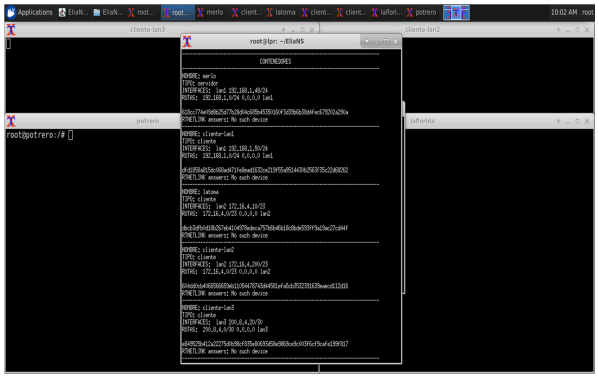

Figura 10: Ejecución de contenedores Docker

5 Conclusiones y Trabajo Futuro

LPR_2.0 logró cumplir con los objetivos propuestos. Se considera que el trabajo realizado mejora de manera significativa no sólo en las prácticas en la asignatura Servicios en SO de Redes, sino también que puede ser usado en cualquier otro ámbito en el cual se requiera experimentar con redes de datos, ampliando así su alcance.

Al ser un desarrollo de software libre, se permite la modificación por otros estudiantes que lo requiera o necesite, adaptándolo en base a otros requisitos específicos.

Si bien LPR_2.0 puede ser ejecutado desde un medio removible como un USB, para poder almacenar las nuevas configuraciones es necesario otro medio de almacenamiento. Se pretende a futuro eliminar esta limitación.

Por otro lado, se está analizando la posibilidad de desarrollar una herramienta educativa para la gestión de memoria debido a que es un tema extensamente desarrollado en cátedras como Introducción a los Sistemas Operativos y Sistemas Operativos de red. El objetivo en un futuro es incluir esta herramienta en LPR_2.0.

Referencias

1. Barrionuevo, M., Gil, C., Giribaldi, M., Suarez, C., y Taffermaberry, C. (2017). “Virtualization in

Education: Portable Network Laboratory". Libro "Communications in Computer and Information Science" (CCIS) vol. 790. Editorial Springer. 2018.
2. Smith, J. E., & Nair, R. (2005). The architecture of virtual machines. Computer, 38(5), 32-38.
3. Chowdhury, N. M. K., & Boutaba, R. (2010). A survey of network virtualization. Computer Networks, 54(5), 862-876.
4. Fernando Rodríguez-Haro, Felix Freitag, Leandro Navarro, A summary of virtualization techniques, Procedia Technology, Volume 3, 2012, Pages 267-272, ISSN 2212-0173
5. A. Souvik Pal , B. Prasant Kumar Pattnaik, Classification of Virtualization Environment for Cloud Computing, Indian Journal of Science and Technology Vol: 6 Issue: 1 January 2013 ISSN:0974-6846
6. Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux Journal, 2014(239), 2.
7. Introducing Execution Drivers and Libcontainer. http://blog.docker.io/2014/03/docker-0-9-introducing-execution-drivers-and-libcontainer/
8. Linux Containers. https://linuxcontainers.org/lxc/introduction/
9. Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S., Elmqvist, N., & Diakopoulos, N. (2016). Designing the user interface: strategies for effective human-computer interaction. Pearson.
10. Turk M. (2001) Perceptual User Interfaces. In: Earnshaw R.A., Guedj R.A., Dam A.., Vince J.A. (eds) Frontiers of Human-Centered Computing, Online Communities and Virtual Environments. Springer, London
11. Riverbank Computing limited: https://riverbankcomputing.com/software/pyqt/intro
12. https://wiki.python.org/moin/TkInter
13. GNS3. Universidad Complutense Madrid: Proyecto de innovación Software libre para ciencias e ingenierías. https://www.ucm.es/pimcd2014-free-software/gns3?fbclid=IwAR2CdQ2FRuIM5zufZqwZDXez_-iGp-XHFD4YKCFtIciOW5WIvniyAqbERPk
14. Dia. (2018). Gnome. https://wiki.gnome.org/Apps/Dia?fbclid=IwAR3kOTQYYuoNrf1VwVWZYAzTsFs8m1XxVpjbX-n31ieYeIXaJ-8npmLxXco
15. KivaNS. (2010). Universidad de Alicante: Aurova. http://www.disclab.ua.es/kiva/
16. LPR download. http://www.dirinfo.unsl.edu.ar/lpr
17. EliaNS. https://github.com/elian10/EliaNS